Effectful Applicative Bisimilarity: Monads, Relators, and Howe's Method

Ugo Dal Lago, Francesco Gavazzo, Paul Blain Levy

Effectful Applicative Bisimilarity: Monads, Relators, and Howe's Method

Ugo Dal Lago and Francesco Gavazzo
Università di Bologna & INRIA Sophia Antipolis
Bologna, Italy

Paul Blain Levy
University of Birmingham
Birmingham, United Kingdom

***Abstract*—We study Abramsky's applicative bisimilarity abstractly, in the context of call-by-value λ-calculi with algebraic effects. We first of all endow a computational λ-calculus with a monadic operational semantics. We then show how the theory of relators provides precisely what is needed to generalise applicative bisimilarity to such a calculus, and to single out those monads and relators for which applicative bisimilarity is a congruence, thus a sound methodology for program equivalence. This is done by studying Howe's method in the abstract.**

I. Introduction

Program equivalence is one of the central notions in the theory of programming languages, and giving satisfactory definitions and methodologies for it is a challenging problem, for example when dealing with higher-order languages. The problem has been approached, since the birth of the discipline, in many different ways. One can define program equivalence through denotational semantics, thus relying on a model and stipulating two programs to be equivalent if and only if they are interpreted by the same denotation. If the calculus at hand is equipped with a notion of *observation*, typically given through some forms of operational semantics, one could proceed following the route traced by Morris [31], and define programs to be *contextually* equivalent when they *behave* the same *in every* context.

Both these approaches have their drawbacks, the first one relying on the existence of a (not too coarse) denotational model, the latter quantifying over all contexts, and thus making concrete proofs of equivalence hard. Handier methodologies for proving programs equivalent have been introduced along the years based on logical relations and applicative bisimilarity. Logical relations were originally devised for typed, normalising languages, but later generalised to more expressive formalisms, e.g., through step-indexing [2] and biorthogonality [5]. Starting from Abramsky's pioneering work on applicative bisimilarity [1], coinduction has also been proved to be a useful methodology for program equivalence, and has been applied to a variety of calculi and language features.

The scenario just described also holds when the underlying calculus is not pure, but effectful. There have been many attempts to study effectful λ-calculi [34], [30] by way of denotational semantics [21], [14], [12], logical relations [6], and applicative bisimilarity [25], [10], [8]. But while the denotational and logical relation semantics of effectful calculi have been studied in the abstract [18], [20], the same cannot be said about applicative bisimilarity and related coinductive techniques. There is a growing body of literature on applicative bisimilarity for calculi with, e.g., nondeterministic [25], and probabilistic effects [10], but each notion of an effect has been studied independently, often getting different results. Distinct proofs of congruence for applicative bisimilarity, even if done through a common methodology, namely the so-called Howe's method [19], do not at all have the same difficulty in each of the cases cited above. As an example, the proof of the so-called Key Lemma relies on duality results from linear programming [38] when done for probabilistic effects, contrarily to the apparently similar case of nondeterministic effects, whose logical complexity is comparable to that for the plain, deterministic λ-calculus [32], [25]. Finally, as the third author observed in his work with Koutavas and Sumii [23], applicative bisimilarity is fragile to the presence of certain effects, like local states or dynamically created exceptions: in these cases, a sort of information hiding is possible which makes applicative bisimilarity simply too weak, and thus unsound for contextual equivalence.

The observations above naturally lead to some questions. Is there any way to factor out the common part of the congruence proof for applicative bisimilarity in the cases above? Where do the limits on the correctness of applicative bisimilarity lie, in presence of effects? The authors strongly believe that the field of coinductive techniques for higher-order program equivalence should be better understood *in the abstract*, this way providing some answers to the questions above, given that generic accounts for effectful λ-calculi abound in the literature [30], [34].

This paper represents a first step towards answering the questions above. We first of all introduce a computational λ-calculus in which general algebraic effects can be represented, and give a monadic operational semantics for it, showing how the latter coincides with the expected one in many distinct concrete examples. We then show how applicative bisimilarity can be defined for any instance of such a monadic λ-calculus, based on the notion of a relator, which allows to account for the possible ways a relation on a set X can be turned into one for TX, where T is a monad. We then single out a set of axioms for monads and relators which allow us to follow Howe's proof of congruence for applicative bisimilarity *in the abstract*. Noticeably, these axioms are satisfied in all the example algebraic effects we consider. The proof of it allows us to understand the deep reasons *why*, say, different instances of Howe's method in the literature seem to have

$$
\begin{aligned}
W &\to V \oplus \mathit{COMP}(V, W)\\
Z &\to T\,\underline{1}\\
T\,\underline{n} &\to (R\,\underline{n}) \oplus (T\,\underline{n+1})\\
R\,\underline{0} &\to \lambda x.x\\
R\,\underline{n+1} &\to \mathit{COMP}(R\,\underline{n}, V)
\end{aligned}
$$

Fig. 1. Two Probabilistic Programs.

$$
\begin{aligned}
W^{\mathsf{raise}} &\to (V \oplus \mathsf{raise}_e) \oplus \mathit{COMP}(V, W^{\mathsf{raise}})\\
Z^{\mathsf{raise}} &\to T\,\underline{1}\\
T\,\underline{n} &\to ((R\,\underline{n}) \oplus \mathsf{raise}_e) \oplus (T\,\underline{n+1})\\
R\,\underline{0} &\to \lambda x.x\\
R\,\underline{n+1} &\to \mathit{COMP}(R\,\underline{n}, V)
\end{aligned}
$$

Fig. 2. Two Probabilistic Programs Throwing Exceptions.

different complexities.

Due to space constraints, many proofs need to be elided. An extended version of this paper with more details is available [9].

II. On Coinduction and Effectful λ-Calculi

In this section, we illustrate how coinduction can be useful when proving the equivalence of programs written in higher-order effectful calculi.

Let us start with a simple example of two supposedly equivalent probabilistic functional programs, W and Z, given in Figure 1. (The expression $\mathit{COMP}(M, N)$ stands for the term $\lambda y.M(Ny)$, and $\oplus$ is a binary operation for fair probabilistic choice.) Both W and Z behave like the n-th composition of a function V with itself with probability $\frac{1}{2^n}$, for every $n > 0$. But how could we even *define* the equivalence of such effectful programs? A natural answer consists in following Morris [31], and stipulate that two programs are contextually equivalent if they behave the same when put in any context, where the observable behaviour of a term can be taken, e.g., as its probability of convergence. Proving two terms to be contextually equivalent can be quite hard, given the universal quantification over all contexts on which contextual equivalence is based.

Applicative bisimilarity is an alternative definition of program equivalence, in which λ-terms are seen as computational objects interacting with their environment by exposing their behaviour, and by taking arguments as input. Applicative bisimilarity has been generalised to effectful λ-calculi of various kinds, and in particular to untyped probabilistic λ-calculi [10]. It is known to be not only a congruence (thus *sound* for contextual equivalence) but also *fully abstract*, at least for call-by-value evaluation [8]. Indeed, applicative bisimilarity can be applied to the example terms in Figure 1, which can this way be proved contextual equivalent (see [9] for the details).

The proof of soundness of applicative bisimilarity in presence of probabilistic effects is significantly more complicated than the original one, although both can be done by following the so-called Howe's method [19]. More specifically, the proof that the Howe extension of similarity is a simulation relies on duality from linear programming (through the Max Flow Min Cut Theorem) when done in presence of probabilistic effects, something that is not required in the plain, deterministic setting, nor in presence of *nondeterministic* choice.

Modern functional programming languages, however, can be "effectful" in quite complex ways. As an example, programs might be allowed not only to evolve probabilistically, but also to have an internal state, to throw exceptions, or to perform some input-output operations. Consider, as another simple example, the programs in Figure 2, a variation on the programs from Figure 1 where we allow an exception e to be thrown by way of the raise_e command. Intuitively, W^{raise} and Z^{raise} behave like W and Z, respectively, but they both throw an exception with a certain probability.

While applicative similarity in presence of dynamically created exceptions is well-known to be unsound [23], the mere presence of the raise_e command does not seem to cause any significant problem. The literature, however, does not offer any result about whether *combining* two or more notions of computational effect for which bisimilarity is known to work well, should be problematic or not. An *abstract* theory accounting for how congruence proofs can be carried out in effectful calculi is simply lacking.

Even if staying within the scope of Howe's method, it seems that each effect between those analysed in the literature is handled by way of some *ad-hoc* notion of bisimulation. As an example, nondeterministic extensions of the λ-calculus can be dealt with by looking at terms as a labelled transition system, while probabilistic extensions of the λ-calculus require a different definition akin to Larsen and Skou's probabilistic bisimulation [10]. What kind of transition system do we need when, e.g., dealing with the example programs from Figure 2? In other words, an abstract theory of effectful applicative bisimilarity would be beneficial from a purely definitional viewpoint, too.

What could come to the rescue here is the analysis of effects and bisimulation which has been carried out in the field of coalgebra [36]. In particular, we here exploit the theory of relators, also known as lax extensions [4], [39].

III. Domains and Monads: Some Preliminaries

In this section, we recall some basic definitions and results on complete partial orders, categories, and monads. Those will be central in the rest of this paper but, due to space constraints, there is no hope to be comprehensive. We refer to this paper's long version [9] or to the many introductory textbooks on

partial order theory [13] or category theory [29] for more details.

A. Domains and Continuous Σ-algebras

We assume basic familiarity with domain theory. In particular, we assume the notions of ω-complete partial order ($\omega\mathbf{CPO}$ for short), and of pointed ω-complete partial order ($\omega\mathbf{CPPO}$ for short), as well as the notions of monotone, continuous, and strict functions, which are standard. Finally, we recall that the category of $\omega\mathbf{CPPO}$s and continuous function is cartesian closed, meaning in particular that $\omega\mathbf{CPPO}$s are closed under (finite) products and exponentials, defined as continuous functions spaces. Following [34], [35], we consider operations (like $\oplus$ or raise_e in the examples from Section II) from a given signature as sources of effects. Semantically, dealing with operation symbols requires the introduction of appropriate algebraic structures interpreting such operation symbols as suitable functions. Combining the algebraic and the order theoretic structures just described, leads to consider algebras carrying a domain structure ($\omega\mathbf{CPPO}$, in this paper), such that *all* function symbols are interpreted as continuous functions. The formal notion capturing all these desiderata is the one of a continuous Σ-algebra [15].

Recall that a signature $\Sigma = (\mathcal{F}, \alpha)$ consists of a set $\mathcal{F}$ of operation symbols and a map $\alpha : \mathcal{F} \to \mathbb{N}$, assigning to each operation symbol a (finite) arity.

Definition 1. *Given a signature Σ, a continuous Σ-algebra is an $\omega\mathbf{CPPO}$ $(D, \sqsubseteq, \bot)$ such that for any function symbol σ in Σ there is an associated continuous function $\sigma^D : D^{\alpha(\sigma)} \to D$.*

Please observe that for a function symbol $\sigma \in \Sigma$, we do not require σ^D to be strict.

Before looking at monads, we now give various examples of concrete algebras which can be given the structure of a continuous Σ-algebra for certain signatures. This testifies the applicability of our theory to a relatively wide range of effects.

Example 1. *Let X be a set: the following are examples of $\omega\mathbf{CPPO}$.*

- *The flat lifting $X_\bot$ of X, defined as $X + \{\bot\}$, ordered as follows: $x \sqsubseteq y$ iff $x = \bot$ or $x = y$.*
- *The set $(X + E)_\bot$ (think to E as a set of exceptions), ordered as in the previous example. We can consider the signature $\Sigma = \{\mathsf{raise}_e \mid e \in E\}$, where each operation symbol raise_e is interpreted as the constant $\mathsf{in}_l(\mathsf{in}_r(e))$.*
- *The powerset $\mathcal{P}X$, ordered by inclusion. The least upper bound of a chain of sets is their union, whereas the bottom is the empty set. We can consider the signature $\Sigma = \{\oplus\}$ containing a binary operation symbol for nondeterministic choice. The latter can be interpreted as (binary) union, which is indeed continuous.*
- *The set of subdistributions $\mathcal{D}X = \{\mu : X \to [0,1] \mid \mathsf{supp}(\mu)$ countable, $\sum_{x\in X} \mu(x) \leq 1\}$ over X, ordered pointwise: $\mu \sqsubseteq \nu$ iff $\forall x \in X.\ \mu(x) \leq \nu(x)$. Note that requiring the support of μ to be countable is equivalent to requiring the existence of $\sum_{x\in X} \mu(x)$. The $\omega\mathbf{CPPO}$ structure is pointwise induced by the one of $[0,1]$ with the natural ordering. The least element is the always zero distribution $x \mapsto 0$ (note that the latter is a subdistribution, and not a distribution). We can consider the signature $\Sigma = \{\oplus_p \mid p \in [0,1]\}$ with a family of probabilistic choice operations indexed by real numbers in $[0,1]$. We can interpret $\oplus_p$ as the binary operation $(x, y) \mapsto p \cdot x + (1-p) \cdot y$, which is indeed continuous.*
- *The set $(S \times X)^S_\bot$, or equivalently $S \rightharpoonup (X \times S)$ (the set of partial states over X) with extension order: $f \sqsubseteq g$ iff $\forall x \in X.\ f(x) \neq \bot \Rightarrow f(x) = g(x)$, for a fixed set S (of states). The bottom element is the totally undefined function $x \mapsto \bot$, whereas the least upper bound of a chain $(f_n)_{n<\omega}$ is computed pointwise. Depending on the choice of S, we can define several continuous operations on $(S \times X)^S_\bot$. For instance, taking $S = \{\mathsf{true}, \mathsf{false}\}$, the set of booleans, we can consider the signature $\Sigma = \{\mathsf{read}, \mathsf{write}_b \mid b \in S\}$ to be interpreted as the continuous operations $read$ and $write_b$ defined by*

$$write_b(f) = x \mapsto f(b);$$
$$read(f, g) = x \mapsto \textit{if } x = \mathtt{true} \textit{ then } f(x) \textit{ else } g(x).$$

B. Monads

The notion of monad is given via the equivalent notion of Kleisli Triple (see [29]). Let $\mathbb{C}$ be a category.

Definition 2. *A Kleisli Triple $\langle T, \eta, (\cdot)^\dagger\rangle$ consists of an endomap T over objects of $\mathbb{C}$, a family of arrows η_X, for any object X, and an operation (called Kleisli extension or Kleisli star) $(\cdot)^\dagger : \mathsf{Hom}_\mathbb{C}(X, TY) \to \mathsf{Hom}_\mathbb{C}(TX, TY)$, (for all objects X, Y) such that $f^\dagger \circ \eta = f$, $\eta^\dagger = id$ and $(g^\dagger \circ f)^\dagger = g^\dagger \circ f^\dagger$ hold, for f and g of the appropriate types.*

We will often denote a Kleisli Triple $\langle T, \eta, (\cdot)^\dagger\rangle$ simply as T. From now on we fix the base category $\mathbb{C}$ to be the category $\mathbb{SET}$ of sets and functions.

Remark 1. *Since we work in $\mathbb{SET}$, we will extensively use the so called bind operator »= in place of Kleisli extensions. Such operator takes as arguments an element u of TX, together with a function $f : X \to TY$ and returns an element u»=f in TY. Concretely, we can define u»=f as $f^\dagger(u)$. Vice versa, we can define the Kleisli extension $f^\dagger$ of f as $x \mapsto (x$»=$f)$.*

Example 2. *All the constructions introduced in Example 1 are monadic. For instance[1]:*

- *The functor $TX = X_\bot$ is (part of) a monad, with left injection as unit and bind operator defined by*

$$u\text{»=}f = \begin{cases} f(x) & \textit{if } u = \mathsf{in}_l(x),\ \textit{for some } x \in X; \\ \mathsf{in}_r(\bot) & \textit{otherwise}. \end{cases}$$

[1]We will be terminologically sloppy, not distinguishing between monads and Kleisli Triples not even on a terminological level. For instance, in the following example we will often speak of functors and monads, although what we really mean is endomaps on objects (of $\mathbb{SET}$) and Kleisli triples.

- *The powerset functor $\mathcal{P}$ is a monad with unit $x \mapsto \{x\}$ and bind operator defined by $u\gg\!=f = \bigcup_{x\in u} f(x)$.*
- *The subdistribution functor $\mathcal{D}$ is a monad with unit given via the Dirac distribution δ and bind operator defined by*

$$\mu\gg\!=f = y \mapsto \sum_{x\in X} \mu(x)\cdot f(x)(y).$$

- *The partiality and exception functor $TX = (X+E)_\perp$ for a given set E of exceptions is a monad with the function $x \mapsto \mathsf{in}_l(\mathsf{in}_l(x))$ as unit. The bind operator is defined by*

$$u\gg\!=f = \begin{cases} u & \text{if } u = \mathsf{in}_r(\perp) \text{ or } u = \mathsf{in}_l(\mathsf{in}_r(e)); \\ f(x) & \text{if } u = \mathsf{in}_l(\mathsf{in}_l(x)). \end{cases}$$

- *The partiality and global state functor $TX = S \to (X \times S)_\perp$ for a given set S of states, is a monad with unit $x \mapsto (s \mapsto (x,s))$ and the bind operator defined by*

$$(\sigma\gg\!=f)(s) = \begin{cases} \perp & \text{if } \sigma(s) = \perp; \\ f(y)(t) & \text{if } \sigma(s) = (y,t). \end{cases}$$

For a given signature Σ, we are interested in monads on $\mathbb{SET}$ that carry a continuous Σ-algebra structure.

Definition 3. *An $\omega\mathbf{CPPO}$ order $\sqsubseteq$ on a monad T is a map that assigns to each set X a relation $\sqsubseteq_X \subseteq TX \times TX$ and an element $\perp_X \in TX$ such that*

- *The structure $(TX, \sqsubseteq_X, \perp_X)$ is an $\omega\mathbf{CPPO}$.*
- *The bind operator is continuous in both arguments. That is,*

$$(\bigsqcup_{n<\omega} u_n)\gg\!=f = \bigsqcup_{n<\omega} (u_n\gg\!=f);$$
$$u\gg\!=(\bigsqcup_{n<\omega} f_n) = \bigsqcup_{n<\omega} (u\gg\!=f_n).$$

We say that $\gg\!=$ is strict in its first argument if we additionally have $\perp\gg\!=f = \perp$ (and similarly for its second argument). We say that T carries a continuous Σ-algebra structure if T has an $\omega\mathbf{CPPO}$ order such that TX is a continuous Σ-algebra with respect to the order $\sqsubseteq_X$, for any set X.

Most of the time we will work with a fixed set X. As a consequence, we will omit subscripts, just writing $\sqsubseteq$ in place of $\sqsubseteq_X$. Similarly, for an operation σ in Σ, we will write σ^T in place of σ^{TX} (the interpretation of σ as an operation on TX).

In the last definition we are essentially regarding the bind operator as a function from $TX \times (X \to TY)$ to TY, continuous in both arguments. This makes sense since $TX \times (X \to TY)$ is an $\omega\mathbf{CPPO}$ (since we can regard any set X as an $\omega\mathbf{CPPO}$ with the identity order, the set $X \to TY$ coincides with the set of continuous functions from X to TX), so that $TX \times (X \to TY)$ is indeed an $\omega\mathbf{CPPO}$, being the product of two $\omega\mathbf{CPPO}$s). Because $\gg\!=$ is continuous in both its arguments, we have $(\bigsqcup_n u_n)\gg\!=(\bigsqcup_n f_n) = \bigsqcup_n (u_n\gg\!=f_n)$.

The bind operation will be useful when giving an operational semantics to the sequential (monadic) composition of programs. As a consequence, although we did not explicitly require the bind operator to be strict (especially in its first argument), such condition will be often desired (especially when giving semantics to call-by-value languages). It is easy to check that all bind operations defined in Example 1 are strict in their first argument.

Remark 2. *The above continuity condition is a special case (since we are in $\mathbb{SET}$) of the more general notion of $\omega\mathbf{CPPO}$-enrichment. Recall that a category $\mathbb{C}$ is $\omega\mathbf{CPPO}$-enriched if each hom-set $\mathsf{Hom}_\mathbb{C}(X,Y)$ carries a partial order $\sqsubseteq$ with an $\omega\mathbf{CPPO}$ structure, and arrows' composition is continuous regarded as a function from $\mathsf{Hom}_\mathbb{C}(X,Y) \times \mathsf{Hom}_\mathbb{C}(Y,Z)$ to $\mathsf{Hom}_\mathbb{C}(X,Z)$. A monad T on $\mathbb{C}$ is order-enriched if the Kleisli category $\mathcal{K}\ell(T)$ is $\omega\mathbf{CPPO}$-enriched. That is, $\mathsf{Hom}_\mathbb{C}(X,TY)$ carries an $\omega\mathbf{CPPO}$-structure such that function composition is continuous and Kleisli star is locally continuous. Concretely, we require the following equations to hold (for arrows f, g of the appropriate type) (cf. [16]):*

$$(\bigsqcup_{n<\omega} f_n)\circ g = \bigsqcup_{n<\omega} (f_n \circ g);$$
$$f^\dagger \circ \bigsqcup_{n<\omega} g_n = \bigsqcup_{n<\omega} (f^\dagger \circ g_n);$$
$$(\bigsqcup_{n<\omega} f_n)^\dagger = \bigsqcup_{n<\omega} f_n^\dagger.$$

Example 3. *Example 1 shows that all monads in Example 2 have an $\omega\mathbf{CPPO}$ order. Moreover, it is just a matter of simple calculations to show that all bind operators are continuous in both arguments.*

IV. A Computational Calculus and Its Operational Semantics

In this section we define a computational λ-calculus. Following [30], [25], [28], we syntactically distinguish between values and computations. We fix a signature Σ of operation symbols (the sources of side-effects), and a monad T carrying a continuous Σ-algebra structure (which describes the nature of the wanted effectful computations generated by the operations in Σ).

Definition 4. *Given a signature Σ, the sets Λ_Σ and $\mathcal{V}_\Sigma$ of terms and values are defined by the following grammars:*

$$M, N \quad ::= \quad \mathsf{return}\ V \mid VW \mid M\ \mathsf{to}\ x.N \mid \sigma(M,\ldots,M);$$
$$V, W \quad ::= \quad x \mid \lambda x.M.$$

where x ranges over a fixed countably infinite set $\mathtt{X}$ of variables and σ ranges over Σ.

The term $(M\ \mathsf{to}\ x.N)$ captures monadic binding (which is usually expressed using a "let-in" notation). A calculus with an explicit separation between terms and values has the advantage to make proofs simpler, without sacrificing expressiveness. For instance, we can encode terms' application MN as $(M\ \mathsf{to}\ x.(N\ \mathsf{to}\ y.xy))$ and vice versa $(M\ \mathsf{to}\ x.N)$ as $(\lambda x.N)M$.

Example 4. *We can model several calculi combining the signatures from Example 1.*

- *For a given set E of exceptions, we can define a probabilistic λ-calculus with exceptions as Λ_Σ, for a signature $\Sigma = \{\oplus_p, \mathsf{raise}_e \mid p \in [0,1], e \in E\}$. In particular, we will have terms of the form $M \oplus_p N$ and raise_e. Replacing the probabilistic choice operator $\oplus_p$ with its nondeterministic counterpart $\oplus$ we obtain a nondeterministic calculus with exceptions.*
- *We can define a nondeterministic calculus with global (boolean) states as Λ_Σ, for a signature $\Sigma = \{\oplus, \mathsf{write}_b, \mathsf{read} \mid b \in \{true, false\}\}$. In particular, we will have terms of the form $M \oplus N$, $\mathsf{write}\ b.M$, and $\mathsf{read}(M,N)$. The intuitive meaning of $\mathsf{write}\ b.M$ is to store b and then continue as M, whereas the intuitive meaning of $\mathsf{read}(M,N)$ is to read the value in the store: if the latter is the boolen true then continue as M, otherwise as N. A formal semantics for these two functions is given in Example 1.*

In what follows, we work with a fixed arbitrary signature Σ. As a consequence, we often denote the sets of terms and values as Λ and $\mathcal{V}$, respectively, thus omitting subscripts. Moreover, we consider terms and values modulo α-equivalence and assume Barendregt Convention [3]. We let $FV(M)$ denote the set of free variables of the term M. A term M is closed if $FV(M) = \emptyset$. We denote finite sets of variables, terms and values using "bar notation": for instance, we write $\bar{x}$ and $\bar{V}$ for a finite set of variables and values, respectively. For a finite set $\bar{x}$ of variables define

$$\Lambda(\bar{x}) = \{M \mid FV(M) \subseteq \bar{x}\};$$
$$\mathcal{V}(\bar{x}) = \{V \mid FV(V) \subseteq \bar{x}\};$$

to be the sets of terms and values with free variables in $\bar{x}$, respectively. The set of closed terms and values are then defined as $\Lambda(\emptyset)$ and $\mathcal{V}(\emptyset)$, and denoted as Λ_0 and $\mathcal{V}_0$, respectively. The term obtained substituting each occurrence of x by V in the term M, denoted $M[x := V]$ can be defined as usual [9].

Big-step semantics associates to each closed term M an element $[\![M]\!]$ in $T\mathcal{V}_0$. Such a semantics is defined by means of an approximation relation $\Downarrow_n$, indexed by a natural number n, whose definition is given in Figure 3. Judgments are of the form $M \Downarrow_n X$, where $M \in \Lambda_0$, $X \in T\mathcal{V}_0$ and $n \geq 0$. Intuitively, a judgment $M \Downarrow_n X$ states that X is the n-th approximation of the computation obtained by call-by-value evaluating M. (By the way, all the results in this paper would remain valid also if evaluating terms in call-by-*name* order, which is however less natural in presence of effects.)

It is easy to see that the set $\{X \in T\mathcal{V}_0 \mid M \Downarrow_n X\}$ of finite approximants of M forms an ω-chain. As a consequence, we can define the evaluation $[\![M]\!]$ of a term M as

$$[\![M]\!] = \bigsqcup_{M \Downarrow_n X} X.$$

This allows us to explicitly capture non-termination (which is usually defined coinductively). For instance, it is easy to show that for the purely (i.e. having no side-effects) divergent program Ω, we have $[\![\Omega]\!] = \bot$. This style of operational semantics

$$\frac{}{M \Downarrow_0 \bot}\ (\mathsf{bot}) \qquad \frac{}{\mathsf{return}\ V \Downarrow_{n+1} \eta(V)}\ (\mathsf{ret})$$

$$\frac{M \Downarrow_n X \qquad N[x := V] \Downarrow_n Y_V}{M\ \mathsf{to}\ x.N \Downarrow_{n+1} X \gg\!= (V \mapsto Y_V)}\ (\mathsf{seq})$$

$$\frac{M[x := V] \Downarrow_n X}{(\lambda x.M)V \Downarrow_{n+1} X}\ (\mathsf{app})$$

$$\frac{M_1 \Downarrow_n X_1 \qquad \dots \qquad M_k \Downarrow_n X_k}{\sigma(M_1, \dots, M_k) \Downarrow_{n+1} \sigma^T(X_1, \dots, X_k)}\ (\mathsf{op})$$

Fig. 3. Big-step Semantics.

[11], [10] is precisely the reason we require the monad T to carry an $\omega\mathbf{CPPO}$ structure. Modelling divergence explicitly turned out to be fundamental in e.g. probabilistic calculi [11].

Since both $\gg\!=$ and σ^T are continuous, we can characterise operational semantics equationally.

Lemma 1. *The following equations hold:*

$$[\![\mathsf{return}\ V]\!] = \eta(V);$$
$$[\![(\lambda x.M)V]\!] = [\![M[x := V]]\!];$$
$$[\![M\ \mathsf{to}\ x.N]\!] = [\![M]\!] \gg\!= (V \mapsto [\![N[x := V]]\!]);$$
$$[\![\sigma(M_1, \dots, M_n)]\!] = \sigma^T([\![M_1]\!], \dots, [\![M_n]\!]).$$

V. On Relational Reasoning

A. Relators

In this section we introduce the concept of relator [39], [27], which is an abstraction meant to capture the possible ways a relation on a set X can be turned into a relation on TX. Recall that for an endofunctor $F : \mathbb{C} \to \mathbb{C}$, an F-coalgebra [36] consists of an object X of $\mathbb{C}$ together with a morphism $\gamma_X : X \to FX$. As usual, we are just concerned with the case in which $\mathbb{C}$ is $\mathbb{SET}$.

Definition 5. *Let F be an endofunctor on $\mathbb{SET}$, and X, Y be sets. A relator Γ for F is a map that associates to each relation $\mathcal{R} \subseteq X \times Y$ a relation $\Gamma\mathcal{R} \subseteq FX \times FY$ such that*

$$=_{FX} \subseteq \Gamma(=_X) \tag{Rel-1}$$
$$\Gamma\mathcal{S} \circ \Gamma\mathcal{R} \subseteq \Gamma(\mathcal{S} \circ \mathcal{R}) \tag{Rel-2}$$
$$\Gamma((f \times g)^{-1}\mathcal{R}) = (Ff \times Fg)^{-1}\Gamma\mathcal{R} \tag{Rel-3}$$
$$\mathcal{R} \subseteq \mathcal{S} \implies \Gamma\mathcal{R} \subseteq \Gamma\mathcal{S} \tag{Rel-4}$$

where for $f : Z \to X$, $g : W \to Y$ we have $(f \times g)^{-1}\mathcal{R} = \{(z, w) \mid f(z)\ \mathcal{R}\ g(w)\}$, and $=_X$ denotes the identity relation on X. A relator Γ is conversive *if $\Gamma(\mathcal{R}^c) = (\Gamma\mathcal{R})^c$, where $\mathcal{R}^c$ denotes the converse of $\mathcal{R}$.*

Example 5. *For each of the monads introduced in previous sections, we give some examples of relators. We use the notation Γ for a relator aimed to capture the structure of a simulation relation, and Δ for a relator aimed to capture the structure of a bisimulation relation. This distinction is*

not formal, and only makes sense in the context of concrete examples: its purpose is to stress that from formal view point, both concrete notions of similarity and bisimilarity are modeled as forms of Γ-similarity (for a suitable relator Γ). Let $\mathcal{R} \subseteq X \times Y$:

- *For the partiality monad $TX = X_\perp$ define the relators $\Gamma_\perp, \Delta_\perp$ by*

$$\begin{aligned} u\ \Gamma_\perp\mathcal{R}\ v \text{ iff } u = \mathsf{in}_l(x) &\implies v = \mathsf{in}_l(y) \wedge x\ \mathcal{R}\ y; \\ u\ \Delta_\perp\mathcal{R}\ v \text{ iff } u = \mathsf{in}_l(x) &\implies v = \mathsf{in}_l(y) \wedge x\ \mathcal{R}\ y, \\ v = \mathsf{in}_l(y) &\implies u = \mathsf{in}_l(x) \wedge x\ \mathcal{R}\ y. \end{aligned}$$

  Note that $u = \mathsf{in}_l(x)$ means, in particular, $u \neq \mathsf{in}_r(\perp)$. Thus, for instance, u and v are $\Gamma_\perp\mathcal{R}$ related if whenever u converges, so does v and the values to which u, v converge are $\mathcal{R}$-related. The relator $\Delta_\perp$ is conversive.
- *For the nondeterministic powerset monad $\mathcal{P}$ define relators $\Gamma_\mathcal{P}$ and $\Delta_\mathcal{P}$ by*

$$\begin{aligned} u\ \Gamma_\mathcal{P}\mathcal{R}\ v \text{ iff } &\forall x \in u.\ \exists y \in v.\ x\ \mathcal{R}\ y; \\ u\ \Delta_\mathcal{P}\mathcal{R}\ v \text{ iff } &\forall x \in u.\ \exists y \in v.\ x\ \mathcal{R}\ y, \\ &\forall y \in v.\ \exists x \in u.\ x\ \mathcal{R}\ y. \end{aligned}$$

  The relator $\Delta_\mathcal{P}$ is conversive.
- *For the probabilistic subdistributions monad $\mathcal{D}$ define relators $\Gamma_\mathcal{D}$ and $\Delta_\mathcal{D}$ by*

$$\begin{aligned} \mu\ \Gamma_\mathcal{D}\mathcal{R}\ \nu \text{ iff } &\forall U \subseteq X.\ \mu(U) \leq \nu(\mathcal{R}(U)); \\ \mu\ \Delta_\mathcal{D}\mathcal{R}\ \nu \text{ iff } &\mu\ \Gamma_\mathcal{D}\mathcal{R}\ \nu\ \wedge\ \nu\ \Gamma_\mathcal{D}\mathcal{R}^c\ \mu; \end{aligned}$$

  where $\mathcal{R}(U) = \{y \in Y \mid \exists x \in U.\ x\ \mathcal{R}\ y\}$ and $\mu(U) = \sum_{x \in U} \mu(x)$. The relator $\Delta_\mathcal{D}$ is conversive.
- *For the exception monad $TX = X + E$ define the relators $\Gamma_\mathcal{E}$ and $\Delta_\mathcal{E}$ by (letters e, e' range over E)*

$$\begin{aligned} u\ \Gamma_\mathcal{E}\mathcal{R}\ v \text{ iff } u = \mathsf{in}_r(e) &\implies v = \mathsf{in}_r(e') \wedge e = e', \\ u = \mathsf{in}_l(x) &\implies v = \mathsf{in}_l(y) \wedge x\ \mathcal{R}\ y; \\ u\ \Delta_\mathcal{E}\mathcal{R}\ v \text{ iff } &u\ \Gamma_\mathcal{E}\mathcal{R}\ v, \\ v = \mathsf{in}_r(e') &\implies u = \mathsf{in}_r(e) \wedge e = e', \\ v = \mathsf{in}_l(y) &\implies u = \mathsf{in}_l(x) \wedge x\ \mathcal{R}\ y. \end{aligned}$$

  The relator $\Delta_\mathcal{E}$ is conversive.
- *For the partiality and exception monad (i.e. the exception monad with divergence) $TX = (X + E)_\perp$ we can define relators simply composing relators for the partiality monad with relators for the exceptions monads (see Lemma 2). Notably, define $\Gamma_{\mathcal{E}_\perp}$ as $\Gamma_\perp \circ \Gamma_\mathcal{E}$ and $\Delta_{\mathcal{E}_\perp}$ as $\Delta_\perp \circ \Delta_\mathcal{E}$. The relator $\Delta_{\mathcal{E}_\perp}$ is conversive.*
- *For the state monad $TX = (X \times S)^S$ define the relator $\Delta_\mathcal{S}$ by*

$$\begin{aligned} f\ \Delta_\mathcal{S}\mathcal{R}\ g \text{ iff } &\forall s \in S.\ s_1 = s_2 \text{ and } x_1\ \mathcal{R}\ x_2, \\ &\text{where } (x_1, s_1) = f(s) \text{ and } (x_2, s_2) = g(s). \end{aligned}$$

  The relator $\Delta_\mathcal{S}$ is conversive.

Checking that the above are indeed relators is a tedious but easy exercise. It is useful to know that the collection of relators is closed under certain operations (see [27] for proofs).

Lemma 2 (Algebra of Relators)**.** *Let F, G be endofunctors on $\mathbb{SET}$. Then:*

1. *The collection of relators for F is closed under arbitrary intersection and converse, where intersection of relators is defined pointwise, and the converse Γ^c of a relator Γ for F is defined by $\Gamma^c(\mathcal{R}) = (\Gamma\mathcal{R}^c)^c$. Moreover, for a relator Γ for F, $\Gamma \cap \Gamma^c$ is the greatest conversive relator contained in Γ.*
2. *Let Γ, Γ' be relators for F, G, respectively. Then $\Gamma' \circ \Gamma$ is a relator for $G \circ F$. Moreover, if both Γ and Γ' are conversive, then so is $\Gamma' \circ \Gamma$.*

We can now give a general notion of (bi)simulation.

B. Bisimulation, in the Abstract

A relator Γ for a monad T expresses the observable part of the side-effects encoded by T. Its abstract nature allows to give abstract definitions of simulation and bisimulation parametric in the notion of observation given by Γ.

Definition 6. *Let $\gamma_X : X \to FX, \gamma_Y : Y \to FY$ be F-coalgebras.*

1. *A Γ-simulation is a relation $\mathcal{R} \subseteq X \times Y$ such that*

$$x\mathcal{R}y \implies \gamma_X(x)\ \Gamma\mathcal{R}\ \gamma_Y(y).$$

2. *Γ-similarity $\precsim^\Gamma_{X,Y}$ is the largest Γ-simulation.*

Example 6. *It is immediate to see that the corresponding notions of Γ-similarity for the (bi)simulation relators of Example 5 coincide with widely used notions of (bi)similarity.*

As usual, the notion of similiarity can be characterised coinductively as the greatest fixed point of a suitable functional.

Definition 7. *Let $\gamma_X : X \to FX$, $\gamma_Y : Y \to FY$ be F-coalgebras. Define the functional $\mathcal{F}^\Gamma_{X,Y} : 2^{X \times Y} \to 2^{X \times Y}$ by*

$$\mathcal{F}^\Gamma_{X,Y}(\mathcal{R}) = (\gamma_X \times \gamma_Y)^{-1}(\Gamma\mathcal{R}).$$

When clear from the context, we will write $\mathcal{F}_\Gamma$ and $\precsim_\Gamma$ in place of $\mathcal{F}^\Gamma_{X,Y}$ and $\precsim^\Gamma_{X,Y}$.

Lemma 3. *The following hold:*

1. *The functional $\mathcal{F}_\Gamma$ is monotone, and thus has a greatest fixed point $\nu\mathcal{F}_\Gamma$.*
2. *A relation $\mathcal{R}$ is a Γ-simulation iff it is a post fixed-point of $\mathcal{F}_\Gamma$. Therefore, Γ-similarity coincides with $\nu\mathcal{F}_\Gamma$.*

It is easy to see that (Rel-1) implies that the identity relation is a Γ-simulation, whereas property (Rel-2) makes Γ-similarity transitive. Moreover, for a conversive relator Γ we have that if $\mathcal{R}$ is a Γ-simulation, then so is $\mathcal{R}^c$. As a consequence, we have the following.

Proposition 1. *Let $\gamma_X : X \to FX$ be an F-coalgebra.*

1. *Γ-similarity is a preorder.*
2. *If Γ is conversive, then Γ-similarity is an equivalence relation.*

Since T is a monad we consider relators that properly interact with the monadic structure of T, which are also known as *lax extensions* for T [4].

Definition 8. *Let T be a monad, X, X', Y, Y' be sets, $f : X \to TX', g : Y \to TY'$ be functions, and $\mathcal{R} \subseteq X \times Y, \mathcal{S} \subseteq X' \times Y'$ be relations. We say that Γ is a relator for T if it is a relator for T regarded as a functor, and*

- $x\ \mathcal{R}\ y \implies \eta_X(x)\ \Gamma\mathcal{R}\ \eta_Y(y)$;
- $u\ \Gamma\mathcal{R}\ v \implies (u \gg\!= f)\ \Gamma\mathcal{S}\ (v \gg\!= g)$, *whenever* $x\ \mathcal{R}\ y \implies f(x)\ \Gamma\mathcal{S}\ g(y)$.

Remark 3. *Definition 8 can be more compactly expressed using Kleisli star, thus requiring that*

$$\mathcal{R} \subseteq (\eta_X \times \eta_Y)^{-1}(\Gamma\mathcal{S}) \qquad \text{(Lax-Unit)}$$

$$\mathcal{R} \subseteq (f \times g)^{-1}(\Gamma\mathcal{S}) \implies \Gamma\mathcal{R} \subseteq (f^\dagger \times g^\dagger)^{-1}(\Gamma\mathcal{S}) \qquad \text{(Lax-Bind)}$$

or diagramatically

$$\begin{array}{ccc} X & \xrightarrow{\mathcal{R}} & Y \\ {\scriptstyle \eta_X}\downarrow & & \downarrow{\scriptstyle \eta_Y} \\ TX & \xrightarrow[\Gamma\mathcal{R}]{} & TY \end{array} \qquad \begin{array}{ccc} X & \xrightarrow{\mathcal{R}} & Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ TX' & \xrightarrow[\Gamma\mathcal{S}]{} & TY' \end{array} \implies \begin{array}{ccc} TX & \xrightarrow{\Gamma\mathcal{R}} & TY \\ {\scriptstyle f^\dagger}\downarrow & & \downarrow{\scriptstyle g^\dagger} \\ TX & \xrightarrow[\Gamma\mathcal{S}]{} & TY \end{array}$$

where we write $\mathcal{R} : X \nrightarrow Y$ for $\mathcal{R} \subseteq X \times Y$.

Example 7. *All relators of the form Γ_T in Example 5 are relators for T. Proving that is quite standard, with the exception of the probabilistic case where the proof essentially relies on the Max Flow Min Cut Theorem [38].*

Definition 9. *Let T come with an $\omega\mathbf{CPPO}$ order $\sqsubseteq$. We say that $\Gamma\mathcal{R}$ is* inductive *if for any ω-chain $(u_n)_{n<\omega}$ in TX, we have:*

$$\bot\ \Gamma\mathcal{R}\ u; \qquad (\omega\text{-comp 1})$$

$$(\forall n.\ u_n\ \Gamma\mathcal{R}\ v) \implies \bigsqcup_n u_n\ \Gamma\mathcal{R}\ v. \qquad (\omega\text{-comp 2})$$

We say that Γ respects *Σ if*

$$(\forall k.\ u_k\ \Gamma\mathcal{R}\ v_k) \implies \sigma(u_1, \ldots, u_n)\ \Gamma\mathcal{R}\ \sigma(v_1, \ldots, v_n) \qquad (\Sigma\text{-comp})$$

for any $\sigma \in \Sigma$, where $k \in \{1, \ldots, \alpha(\sigma)\}$.

Remark 4. *For a monad T carrying a continuous Σ-algebra structure and a function $f : X \to TY$, we required $f^\dagger : TX \to TY$ to be continuous, TX being an $\omega\mathbf{CPPO}$. Since TX is also a Σ-algebra, it seems natural to require $f^\dagger$ to be also a Σ-algebra homomorphism. In fact, such requirement implies condition (Σ-comp) and has the advantage of being more general than the latter, not depending from the specific relator considered (see [9] for more details). To the ends of this paper, condition (Σ-comp) is sufficient and thus we will use that throughout.*

Following Abramsky [1] we introduce Applicative Transition System (ATSs) over a monad (taking into account effectful computations) and define the notion of *applicative simulation*. Let T be a monad.

Definition 10. *An applicative transition system (over T) consists of the following:*

- *A state space made of a pair of sets (X, Y) modelling closed terms and values, respectively.*
- *An evaluation function $\varepsilon : X \to TY$.*
- *An application function $\cdot : Y \to Y \to X$.*

The notion of ATS distinguishes between terms and values. As a consequence, we often deal with pairs of relations $(\mathcal{R}_X, \mathcal{R}_Y)$, where $\mathcal{R}_X, \mathcal{R}_Y$ are relations over X and Y, respectively. We refer to such pairs as XY-relations. XY-relations belongs to $2^{X\times X} \times 2^{Y\times Y}$. The latter, being the product of complete lattices, is itself a complete lattice.

Definition 11. *Let Γ be a relator for T. An applicative Γ-simulation is an XY-relation $\mathcal{R} = (\mathcal{R}_X, \mathcal{R}_Y)$ such that:*

$$x\ \mathcal{R}_X x' \implies \varepsilon(x)\ \Gamma\mathcal{R}_Y\ \varepsilon(x') \qquad \text{(Sim-1)}$$

$$y\ \mathcal{R}_Y y' \implies \forall w \in Y.\ y \cdot w\ \mathcal{R}_X\ y' \cdot w \qquad \text{(Sim-2)}$$

The above definition induces an operator $\mathcal{B}_\Gamma$ on $2^{X\times X} \times 2^{Y\times Y}$ defined for $\mathcal{R} = (\mathcal{R}_X, \mathcal{R}_Y)$ as $(\mathcal{B}_\Gamma(\mathcal{R}_X), \mathcal{B}_\Gamma(\mathcal{R}_Y))$, where

$$\mathcal{B}_\Gamma(\mathcal{R}_X) = \{(x, x') \mid \varepsilon(x)\ \Gamma\mathcal{R}_Y\ \varepsilon(x')\};$$
$$\mathcal{B}_\Gamma(\mathcal{R}_Y) = \{(y, y') \mid \forall w \in Y.\ y \cdot w\ \mathcal{R}_X\ y' \cdot w\}.$$

It is easy to prove that since Γ is monotone, then so is $\mathcal{B}_\Gamma$. As a consequence, we can define applicative Γ-similarity as the greatest fixed point $\nu\mathcal{B}_\Gamma$ of $\mathcal{B}_\Gamma$.

Proposition 2. *The following hold:*

1. *Applicative Γ-similarity $\nu\mathcal{B}_\Gamma$ is a preorder.*
2. *If Γ is conversive, then $\nu\mathcal{B}_\Gamma$ is an equivalence relation.*

VI. Contextual Preorder and Applicative Similarity

In the previous section, the axioms needed to generalise applicative bisimilarity to our setting have been given. What remains to be done is to appropriately instantiate all this to Λ_Σ. We introduce the notions of contextual preorder and applicative similarity (which will be then extended to contextual equivalence and applicative *bi*similarity). From now we assume to have a monad T carrying a continuous Σ-algebra structure. Moreover, we assume any relator for T to be inductive and to respect Σ. It is convenient to work with generalisations of relations on closed terms (resp. values) called λ-term relations.

Definition 12. *An open relation over terms is a set $\mathcal{R}_\Lambda$ of triples $(\bar{x}, M, N)$ where $M, N \in \Lambda(\bar{x})$. Similarly, an open relation over values is a set $\mathcal{R}_\mathcal{V}$ of triples $(\bar{x}, V, W)$ where $V, W \in \mathcal{V}(\bar{x})$. A λ-term relation is a pair $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ made of an open relation $\mathcal{R}_\Lambda$ over terms and an open relation $\mathcal{R}_\mathcal{V}$ over values. A closed λ-term relation is a pair $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ where $\mathcal{R}_\Lambda \subseteq \Lambda_0 \times \Lambda_0$ and similarly for $\mathcal{R}_\mathcal{V}$.*

$$\forall x \in \bar{x}.\ \bar{x} \vdash x\ \mathcal{R}_{\mathcal{V}}\ x \tag{Comp1}$$
$$\forall x \notin \bar{x}.\ \bar{x} \cup \{x\} \vdash M\ \mathcal{R}_{\Lambda}\ N \implies \bar{x} \vdash \lambda x.M\ \mathcal{R}_{\mathcal{V}}\ \lambda x.N \tag{Comp2}$$
$$\bar{x} \vdash V\ \mathcal{R}_{\mathcal{V}}\ W \implies \bar{x} \vdash \mathsf{return}\ V\ \mathcal{R}_{\Lambda}\ \mathsf{return}\ W \tag{Comp3}$$
$$\bar{x} \vdash V\ \mathcal{R}_{\mathcal{V}}\ V' \wedge \bar{x} \vdash W\ \mathcal{R}_{\mathcal{V}}\ W' \implies \bar{x} \vdash VW\ \mathcal{R}_{\Lambda}\ V'W' \tag{Comp4}$$
$$\forall x \notin \bar{x}.\ \bar{x} \vdash M\ \mathcal{R}_{\Lambda}\ M' \wedge \bar{x} \cup \{x\} \vdash N\ \mathcal{R}_{\Lambda}\ N' \implies \bar{x} \vdash (M\ \mathsf{to}\ x.N)\ \mathcal{R}_{\Lambda}\ (M'\ \mathsf{to}\ x.N') \tag{Comp5}$$
$$\bar{x} \vdash M_1\ \mathcal{R}_{\Lambda}\ N_1 \wedge \cdots \wedge \bar{x} \vdash M_{\alpha(\sigma)}\ \mathcal{R}_{\Lambda}\ N_{\alpha(\sigma)} \implies \bar{x} \vdash \sigma(M_1, \ldots, M_{\alpha(\sigma)})\ \mathcal{R}_{\Lambda}\ \sigma(N_1, \ldots, N_{\alpha(\sigma)}) \tag{Comp6}$$

Fig. 4. Compatibility Clauses.

Remark 5. *Formally, we can see an open relation over terms (and similarly over values) as an element of the cartesian product $\prod_{\bar{x}} 2^{\Lambda(\bar{x}) \times \Lambda(\bar{x})}$. That is, an open relation is a function that associates to each finite set $\bar{x}$ of variables a (binary) relation between open terms in $\Lambda(\bar{x})$. Since, $2^{\Lambda(\bar{x}) \times \Lambda(\bar{x})}$ is a complete lattice, for any finite set of variables $\bar{x}$, then so is $\prod_{\bar{x}} 2^{\Lambda(\bar{x}) \times \Lambda(\bar{x})}$. That is, the set of open relations over terms (and over values) forms a complete lattice (the order is given pointwise). As a consequence, the set of λ-term relations is a complete lattice as well. These algebraic properties allow us to define open relations both inductively and coinductively, and, in particular, to extend notions and results developed in the relational calculus [25], [24], [17], [26].*

We will use infix notation and write $\bar{x} \vdash M\ \mathcal{R}_{\Lambda}\ N$ to indicate that $(\bar{x}, M, N) \in \mathcal{R}_{\Lambda}$. The same convention applies to values and open relations over values. For a λ-term relation $\mathcal{R} = (\mathcal{R}_{\Lambda}, \mathcal{R}_{\mathcal{V}})$, we often write $\bar{x} \vdash M\ \mathcal{R}\ N$ (i.e. $(\bar{x}, M, N) \in \mathcal{R}$) for $\bar{x} \vdash M\ \mathcal{R}_{\Lambda}\ N$ (i.e. $(\bar{x}, M, N) \in \mathcal{R}_{\Lambda}$). The same convention holds for values and $\mathcal{R}_{\mathcal{V}}$. Finally, we will use the notations $\emptyset \vdash M\ \mathcal{R}\ N$ and $M\ \mathcal{R}\ N$ interchangeably (and similarly for values).

There is a canonical way to extend a closed relation to an open one.

Definition 13. *Define the open extension operator mapping a closed relation over terms $\mathcal{R}$ to the open relation $\mathcal{R}^{\circ}$ (over terms) as follows: $(\bar{x}, M, N) \in \mathcal{R}^{\circ}$ iff $M, N \in \Lambda(\bar{x})$, and for all $\bar{V}$, $M[\bar{x} := \bar{V}]\ \mathcal{R}\ N[\bar{x} := \bar{V}]$ holds.*

The notion of open extension for a closed relation over values can be defined in a similar way (using the appropriate notion of substitution).

The notion of reflexivity, symmetry and transitivity straightforwardly extends to open λ-term relation (see e.g. [33]).

Definition 14. *Let $\mathcal{R} = (\mathcal{R}_{\Lambda}, \mathcal{R}_{\mathcal{V}})$ be a λ-term relation. We say that $\mathcal{R}$ is* compatible *if the clauses in Figure 4 hold. We say that $\mathcal{R}$ is a* precongruence *if it is a compatible preorder. We say that $\mathcal{R}$ is a* congruence *if it is a compatible equivalence.*

It is useful to characterise compatible relations via the notion of compatible refinement.

Definition 15. *Let $\mathcal{R} = (\mathcal{R}_{\Lambda}, \mathcal{R}_{\mathcal{V}})$ be a λ-term relation. Define the compatible refinement $\widehat{\mathcal{R}}$ of $\mathcal{R}$ as the pair $(\widehat{\mathcal{R}}_{\Lambda}, \widehat{\mathcal{R}}_{\mathcal{V}})$, where $\mathcal{R}_{\Lambda}$ and $\mathcal{R}_{\mathcal{V}}$ are inductively defined by rules in Figure 5. It is easy to see that a λ-term relation $\mathcal{R}$ is compatible iff $\widehat{\mathcal{R}} \subseteq \mathcal{R}$ holds.*

$$\frac{}{\bar{x} \vdash x\ \widehat{\mathcal{R}}_{\mathcal{V}}\ x}\ x \in \bar{x} \qquad \frac{\bar{x} \cup \{x\} \vdash M\ \mathcal{R}_{\Lambda}\ N}{\bar{x} \vdash \lambda x.M\ \widehat{\mathcal{R}}_{\mathcal{V}}\ \lambda x.N}\ x \notin \bar{x}$$

$$\frac{\bar{x} \vdash V\ \mathcal{R}_{\mathcal{V}}\ W}{\bar{x} \vdash \mathsf{return}\ V\ \widehat{\mathcal{R}_{\Lambda}}\ \mathsf{return}\ W}$$

$$\frac{\bar{x} \vdash V\ \mathcal{R}_{\mathcal{V}}\ V' \qquad \bar{x} \vdash W\ \mathcal{R}_{\mathcal{V}}\ W'}{\bar{x} \vdash VW\ \widehat{\mathcal{R}}_{\Lambda}\ V'W'}$$

$$\frac{\bar{x} \vdash M\ \mathcal{R}_{\Lambda}\ M' \qquad \bar{x} \cup \{x\} \vdash N\ \mathcal{R}_{\Lambda}\ N'}{\bar{x} \vdash M\ \mathsf{to}\ x.N\ \widehat{\mathcal{R}}_{\Lambda}\ M'\ \mathsf{to}\ x.N'}\ x \notin \bar{x}$$

$$\frac{\bar{x} \vdash M_1\ \mathcal{R}_{\Lambda}\ N_1 \qquad \ldots \qquad \bar{x} \vdash M_n\ \mathcal{R}_{\Lambda}\ N_n}{\bar{x} \vdash \sigma(M_1, \ldots, M_n)\ \widehat{\mathcal{R}}_{\Lambda}\ \sigma(N_1, \ldots, N_n)}$$

Fig. 5. Compatible Refinement Rules.

The above notion of precongruence can be justified by observing that when a relation $\mathcal{R}$ is a preorder, being a precongruence does exactly mean to be closed under the term constructors of the language. That could be formally expressed by saying that $\mathcal{R}$ is a precongruence if and only if $\bar{x} \vdash M\ \mathcal{R}\ N$ implies $\bar{x} \vdash C[M]\ \mathcal{R}\ C[N]$, for any term context $C[\cdot]$. Defining term contexts requires some care. In particular, when dealing with the contextual preorder it is not possible to reason modulo α-conversion, thus making definition syntactically involved (see [25], [24], [33] for details). As remarked in [33], it is possible to avoid those difficulties by giving a coinductive characterisation of the contextual preorder in the style of [25], [17]. Essentially, the contextual preorder (and, similarly the contextual equivalence) is defined as the largest compatible and preadequate (see Definition 16) λ-term relation. It is then easy to provide a more syntactic definition of contextual preorder and to prove that the two given definitions are equivalent [17], [25], [33].

The notion of adequacy defines the available observation on values. Being in an untyped setting, it is customary not to observe them.

Definition 16. *Let $\mathcal{U}$ denote $\mathcal{V}_0 \times \mathcal{V}_0$ seen as a closed relation, i.e. the trivial relation relating all values. We say that a relation $\mathcal{R}$ on terms is preadequate if*

$$\emptyset \vdash M\ \mathcal{R}\ N \implies [\![M]\!]\ \Gamma\mathcal{U}\ [\![N]\!];$$

where $M, N \in \Lambda_0$. That is, a relation $\mathcal{R}$ on terms is preadequate if whenever $\mathcal{R}$ relates two closed terms, evaluating these programs produces the same side-effects. A λ-term relation $\mathcal{R} = (\mathcal{R}_{\Lambda}, \mathcal{R}_{\mathcal{V}})$ is preadequate iff $\mathcal{R}_{\Lambda}$ is.

Example 8. *It is easy to check that the above notion of adequacy (together with the relators in Example 5) captures standard notions of adequacy used for untyped λ-calculi.*

- *Consider a calculus without operation symbols and with operational semantics over $(\mathcal{V}_0)_\perp$. A relation is preadequate if whenever $\emptyset \vdash M\ \mathcal{R}\ N$, then if M converges, then so does N.*
- *Consider a nondeterministic calculus with operational semantics over $\mathcal{PV}_0$. A relation is preadequate if whenever $\emptyset \vdash M\ \mathcal{R}\ N$, then if there exists a value V to which M may converge (i.e. $V \in [\![M]\!]$), then there exists a value W to which N may converge (i.e. $W \in [\![N]\!]$).*
- *Consider a probabilistic calculus with operational semantics over $\mathcal{DV}_0$. A relation is preadequate if whenever $\emptyset \vdash M\ \mathcal{R}\ N$, then the probability of convergence of M is smaller or equal than the probability of convergence of N.*

Following [25], we shall define the Γ-*contextual preorder* as the largest λ-term relation that is both compatible and preadequate.

Definition 17. *Let $\mathbb{CA}$ be the set of relations on terms that are both compatible and preadequate. Then define $\leq_\Gamma$ as $\bigcup \mathbb{CA}$.*

Proposition 3. *The Γ-contextual preorder $\leq_\Gamma$ is a compatible and preadequate preorder.*

Finally, we can define the notion of an applicative simulation observing how the collection of closed terms and values, together with the operational semantics defined in previous section, carry an ATS structure.

Definition 18. *A closed relation $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ respects values if for all closed values V, W, $V\ \mathcal{R}_\mathcal{V}\ W$ implies $VU\ \mathcal{R}_\Lambda\ WU$, for any closed value U.*

We can now define an ATS over closed λ-terms taking the pair $(\Lambda_0, \mathcal{V}_0)$ as state space and the map $[\![\cdot]\!] : \Lambda_0 \to T\mathcal{V}_0$ as evaluation function. Instantiating the general definition of applicative Γ-simulation we obtain:

Definition 19. *Let Γ be a relator for the monad T. A closed relation $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ is an applicative Γ-simulation if:*

- $M\ \mathcal{R}_\Lambda\ N \implies [\![M]\!]\ \Gamma\mathcal{R}_\mathcal{V}\ [\![N]\!]$;
- *$\mathcal{R}$ respects values.*

We can then define applicative Γ-similarity $\precsim_\Gamma$ as the largest applicative Γ-simulation, which we know to be a preorder by Proposition 2. Most of the time the relator Γ will be fixed; in those cases we will often write $\precsim$ in place of $\precsim_\Gamma$.

Example 9. *It is immediate to see that using the relators in Example 5 we recover well-known notions of simulation and bisimulation.*

We want to prove that applicative similarity is a sound proof technique for contextual preorder. That is, we want to prove that $\precsim_\Gamma \subseteq \leq_\Gamma$ holds. The relation $\leq_\Gamma$ being defined as the largest preadequate compatible relation, the above inclusion is established by proving that $\precsim_\Gamma$ is a precongruence.

VII. Howe's Method and Its Soundness

In this section we generalise Howe's technique to show that applicative similarity is a precongruence, thus a sound proof technique for the contextual preorder. Our generalisation shows how Howe's method crucially (but only!) depends on the structure of the monad modelling side-effects and the relators encoding their associated notion of observation.

Definition 20. *Let $\mathcal{R}$ be a closed λ-term relation. The Howe extension $\mathcal{R}^H$ of $\mathcal{R}$ is defined as the least relation $\mathcal{S}$ such that $\mathcal{S} = \mathcal{R}^\circ \circ \widehat{\mathcal{S}}$.*

It was observed in [26] that the above equation actually defines a unique relation.

Lemma 4. *Let $\mathcal{R}$ be a closed λ-term relation. Then there is a unique relation $\mathcal{S}$ such that $\mathcal{S} = \mathcal{R}^\circ \circ \widehat{\mathcal{S}}$.*

As a consequence, $\mathcal{R}^H$ can be characterised both inductively and coinductively.

Here we give the well-known inductive characterisations of $\mathcal{R}^H$ as the pair $(\mathcal{R}^H_\Lambda, \mathcal{R}^H_\mathcal{V})$ inductively defined by rules in Figure 6.

The following lemma states some nice properties of Howe's lifting of preorder relations. The proof is standard and can be found in, e.g., [10].

Lemma 5. *Let $\mathcal{R}$ be a preorder. The following hold:*
1. *$\mathcal{R} \circ \mathcal{R}^H \subseteq \mathcal{R}^H$.*
2. *$\mathcal{R}^H$ is compatible, and thus reflexive.*
3. *$\mathcal{R} \subseteq \mathcal{R}^H$.*

We now consider the Howe extension $\precsim^H_\Gamma$ of applicative Γ-similarity. Since $\precsim_\Gamma$ is a preorder (Proposition 2), $\precsim^H_\Gamma$ is a compatible relation containing $\precsim_\Gamma$.

Definition 21. *A λ-term relation $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ is value-substitutive if $x \vdash M\ \mathcal{R}_\Lambda\ N$ and $\emptyset \vdash V\ \mathcal{R}_\mathcal{V}\ W$ imply $\emptyset \vdash M[x := V]\ \mathcal{R}_\Lambda\ N[x := W]$.*

Lemma 6. *The relation $\precsim^H_\Gamma$ is value-substitutive.*

Summing up, we have defined a compatible relation $\precsim^H_\Gamma$ which is value-substitutive and contains $\precsim_\Gamma$. As a consequence, to prove that the latter is compatible it is sufficient to prove $\precsim^H_\Gamma \subseteq \precsim_\Gamma$. We can proceed coinductively, showing that $\precsim^H_\Gamma$ is an applicative Γ-simulation. This is proved via the so-called Key Lemma. It is useful to spell out basic facts on the Howe extension of applicative similarity that we will extensively use. Let Γ be a relator (in the following we assume to have fixed a relator Γ, thus omitting subscripts).

Lemma 7. *Let Γ be a fixed relator and write $\precsim$ for $\precsim_\Gamma$. Then*
1. *$\precsim \circ \precsim^H \subseteq \precsim^H$.*
2. *$(\Gamma\ \precsim) \circ (\Gamma\ \precsim^H) \subseteq \Gamma\ \precsim^H$.*

We can now state and prove the Key Lemma.

Lemma 8 (Key Lemma)**.** *Let $\precsim^H = (\precsim^H_\Lambda, \precsim^H_\mathcal{V})$ be the Howe extension of applicative similarity. If $\emptyset \vdash M\ \precsim^H_\Lambda\ N$ and $M \Downarrow_n X$, then $X\ \Gamma\ \precsim^H_\mathcal{V}\ [\![N]\!]$.*

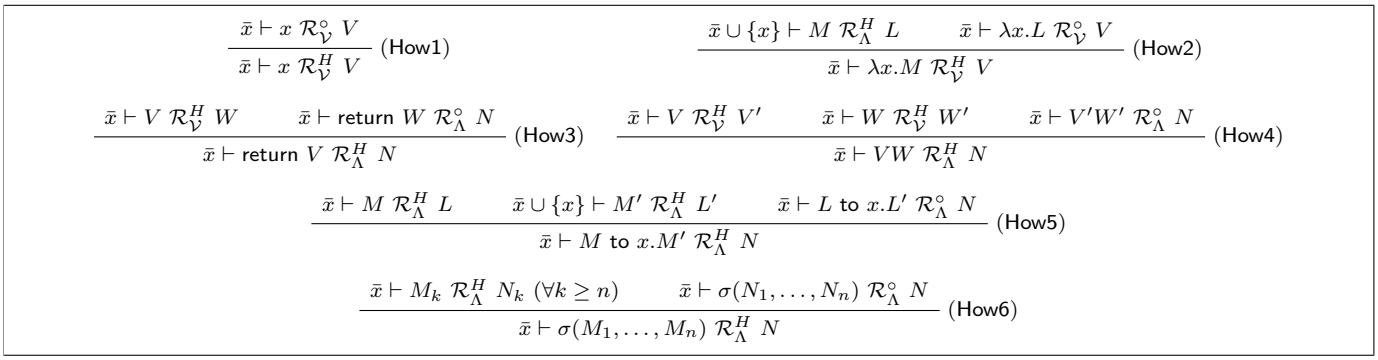

Fig. 6. Howe's Extension Rules.

Proof. The proof proceeds by induction on the derivation of the judgment $M \Downarrow_n X$. Some interesting cases follow:

Case (app). Suppose the judgment $(\lambda x.M)V \Downarrow_{n+1} X$ has been obtained from the judgment $M[x := V] \Downarrow_n X$. By hypothesis we have $\emptyset \vdash (\lambda x.M)V \precsim^H_\Lambda N$, meaning that the latter must have been obtained as the conclusion of an instance of (How4). We thus obtain $\emptyset \vdash \lambda x.M \precsim^H_{\mathcal{V}} W$, $\emptyset \vdash V \precsim^H_{\mathcal{V}} U$ and $WU \precsim_\Lambda N$, for values W, U. Looking at the first of these three judgments, we see that it must be the conclusion of an instance of rule (How2). Therefore, we have $\{x\} \vdash M \precsim^H_\Lambda L$ and $\lambda x.L \precsim_{\mathcal{V}} W$. Since $\precsim^H$ is value-substitutive, from $\{x\} \vdash M \precsim^H_\Lambda L$ and $\emptyset \vdash V \precsim^H_{\mathcal{V}} U$ we conclude $\emptyset \vdash M[x := V] \precsim^H_\Lambda L[x := U]$. We can now apply the induction hypothesis on the latter and $M[x := V] \Downarrow_n X$, obtaining $X\ \Gamma(\precsim^H_{\mathcal{V}})\ [\![L[x := U]]\!]$. Since $\precsim$ respects values, from $\lambda x.L \precsim_{\mathcal{V}} W$ we infer $(\lambda x.L)U \precsim_\Lambda WU$, which gives, by very definition of applicative similarity, $[\![(\lambda x.L)U]\!]\ \Gamma(\precsim_{\mathcal{V}})\ [\![WU]\!]$. By Lemma 1, $[\![(\lambda x.L)U]\!] = [\![L[x := U]]\!]$, and thus, $X\ \Gamma(\precsim^H_{\mathcal{V}})\ [\![WU]\!]$, by Lemma 7. Finally, from $WU \precsim_{\mathcal{V}} N$ we obtain $[\![WU]\!]\ \Gamma(\precsim_{\mathcal{V}})\ [\![N]\!]$, which allows us to conclude $X\ \Gamma(\precsim_{\mathcal{V}})\ [\![N]\!]$ by Lemma 7.

Case (seq). Suppose the judgment $(M\ \mathsf{to}\ x.M') \Downarrow_{n+1} X \gg\!= (V \mapsto Y_V)$ has been obtained from $M \Downarrow_n X$ and $M'[x := V] \Downarrow_n Y_V$. By hypothesis we have $\emptyset \vdash M\ \mathsf{to}\ x.M' \precsim^H_\Lambda N$, which must have been obtained via an instance of rule (How5) thus giving $\emptyset \vdash M \precsim^H_\Lambda L$, $\{x\} \vdash M' \precsim^H_\Lambda L'$ and $\emptyset \vdash L\ \mathsf{to}\ x.L' \precsim_\Lambda N$. We can apply the induction hypothesis on $M \Downarrow_n X$ and $\emptyset \vdash M \precsim^H_\Lambda L$ obtaining $X\ \Gamma(\precsim^H_{\mathcal{V}})\ [\![L]\!]$. We now claim to have

$$X \gg\!= (V \mapsto Y_V)\ \Gamma(\precsim^H_{\mathcal{V}})\ [\![L]\!] \gg\!= (V \mapsto [\![L'[x := V]]\!]).$$

The latter is equal to $[\![L\ \mathsf{to}\ x.L']\!]$, by Lemma 1. Besides, $\emptyset \vdash L\ \mathsf{to}\ x.L' \precsim_\Lambda N$ entails $[\![L\ \mathsf{to}\ x.L']\!]\ \Gamma(\precsim_{\mathcal{V}})\ [\![N]\!]$: we conclude $X \gg\!= (V \mapsto Y_V)\ \Gamma(\precsim^H_{\mathcal{V}})\ [\![N]\!]$, by Lemma 7.

The above claim directly follows from (Lax-Bind). In fact, since $X\ \Gamma(\precsim^H_{\mathcal{V}})\ [\![L]\!]$ holds, by (Lax-Bind) it is sufficient to prove that $V \precsim^H_{\mathcal{V}} W$ implies $Y_V\ \Gamma(\precsim^H_{\mathcal{V}})\ [\![L'[x := W]]\!]$. Assume $V \precsim^H_{\mathcal{V}} W$, i.e. $\emptyset \vdash V \precsim^H_{\mathcal{V}} W$. The latter, together with $\{x\} \vdash M' \precsim^H_\Lambda L'$, implies $\emptyset \vdash M'[x := V] \precsim^H_\Lambda L'[x := W]$, since $\precsim^H$ is value-substitutive. We can finally apply the inductive hypothesis on the latter and $M'[x := V] \Downarrow_n Y_V$, thus concluding the wanted thesis.

□

Corollary 1. *The relation $\precsim^H_\Gamma$ is an applicative Γ-simulation.*

Proof. Suppose $M \precsim^H_\Lambda N$. We have to prove $[\![M]\!]\ \Gamma \precsim^H_{\mathcal{V}} [\![N]\!]$, i.e. $\bigsqcup_{M \Downarrow_n X} X\ \Gamma \precsim^H_{\mathcal{V}} [\![N]\!]$. The latter follows from (ω-comp 1) by the Key Lemma. Finally, since $\precsim^H$ is compatible, it clearly respects values. □

Theorem 1. *Similarity is a precongruence. Moreover, it is sound for contextual preorder $\leq_\Gamma$.*

Proof. We already know $\precsim_\Gamma$ is a preorder. By previous corollary it follows that $\precsim_\Gamma$ coincides with $\precsim^H_\Gamma$, so that $\precsim_\Gamma$ is also compatible, and thus a precongruence. Now for soundness. We have to prove $\precsim_\Gamma \subseteq \leq_\Gamma$. Since $\leq_\Gamma$ is defined as the largest preadequate compatible relation, it is sufficient to prove that $\precsim_\Gamma$ is preadequate (we have already showed it is compatible), which directly follows from Sim-1, since the universal relation $\mathcal{U}$ contains $\precsim_{\mathcal{V}}$. □

VIII. Bisimilarity, Two-similarity and Contextual Equivalence

In this section we extend previous definitions and results to come up with sound proof techniques for contextual *equivalence*. In particular, by observing that contextual equivalence always coincides with the intersection between the contextual preorder and its converse, Theorem 1 implies that two-way similarity (i.e. the intersection between applicative similarity and its converse) is contained in contextual equivalence. Applicative bisimilarity being finer than two-way similarity, we can also conclude the former to be a sound proof technique for contextual equivalence.

Given a relator Γ, we can extract a canonical notion of Γ-bisimulation from the one of Γ-simulation following the idea that a bisimulation is a relation $\mathcal{R}$ such that both $\mathcal{R}$ and $\mathcal{R}^c$ are simulations. Recall that given a relator Γ we can define a converse operation Γ^c as $\Gamma^c(\mathcal{R}) = (\Gamma(\mathcal{R}^c))^c$. Γ^c is indeed a relator. Similarly, we have proved that the intersection of relators is again a relator.

Definition 22 (Γ-bisimulation)**.** *Given a relator Γ, we say that a relation $\mathcal{R}$ is a Γ-bisimulation if it is a $(\Gamma \cap \Gamma^c)$-simulation.*

From the above definition, it directly follows that a relation $\mathcal{R}$ is a Γ-bisimulation if and only if both $\mathcal{R}$ and $\mathcal{R}^c$ are Γ-simulation.

Since, by Lemma 2, $\Gamma \cap \Gamma^c$ is a relator, we can define Γ-bisimilarity $\sim_\Gamma$ as $(\Gamma \cap \Gamma^c)$-similarity. It is easy to see that $\sim_\Gamma$ is an equivalence relation (note that if $\mathcal{R}$ is a Γ-simulation, then so is Γ^c).

Definition 23. *Let Γ be a relator. Define Γ-cosimilarity $\gtrsim_\Gamma$ as $(\precsim_\Gamma)^c$. Define Γ two-way similarity $\simeq_\Gamma$ as $\precsim_\Gamma \cap \gtrsim_\Gamma$.*

As usual, bisimilarity is finer than two-way similarity, meaning that $\sim_\Gamma \subseteq \simeq_\Gamma$. Moreover, taking Γ to be the simulation relator for the powerset monad (see Example 5), we have that $\sim_\Gamma$ and $\simeq_\Gamma$ do not coincide. See e.g. [25], [33].

Recall that we have defined the Γ-contextual preorder $\leq_\Gamma$ as the largest relation that is both compatible and Γ-preadequate. In analogy with what we did for simulation and bisimulation we can give the following:

Definition 24. *Let Γ be a relator. Define Γ-contextual equivalence $\equiv_\Gamma$ as the largest relation that is both compatible and $(\Gamma \cap \Gamma^c)$-preadequate. That is, define $\equiv_\Gamma$ as $\leq_{\Gamma \cap \Gamma^c}$.*

Although bisimilarity is finer than two-way similarity, this is not the case for contextual equivalence and the associated contextual preorders.

Proposition 4. *Let Γ be a relator. Then, $\equiv_\Gamma = \leq_\Gamma \cap \geq_\Gamma$.*

We can finally prove our soundness result.

Theorem 2 (Soundness)**.** *Let Γ be a relator. Two-way similarity $\simeq_\Gamma$ is a congruence, and thus sound for contextual equivalence $\equiv_\Gamma$. Since bisimilarity $\sim_\Gamma$ is finer than $\simeq_\Gamma$, it is sound for $\equiv_\Gamma$ as well.*

Proof. From Theorem 1 we know that $\precsim_\Gamma$ is a precongruence and that $\precsim_\Gamma \subseteq \leq_\Gamma$. It follows $\gtrsim_\Gamma$ is a precongruence as well, and that $\gtrsim_\Gamma \subseteq \geq_\Gamma$ holds. We can conclude $\simeq_\Gamma$ is a congruence and $\simeq_\Gamma \subseteq \equiv_\Gamma$. Since $\sim_\Gamma \subseteq \simeq_\Gamma$, we also have $\sim_\Gamma \subseteq \equiv_\Gamma$. □

Noticeably, Theorem 2 can be seen as a proof of soundness for applicative bisimilarity in any calculus Λ_Σ which respects our requirements (see Definition 8, 9), and in particular for those described in Example 5. The case of probabilistic calculi is illuminating: the apparent complexity of all proofs of congruence from the literature [10], [8] has been confined to the proof that the relator for subdistributions satisfies our axiomatics.

Theorem 2 also allows us to prove W^{raise} and Z^{raise}, our example programs from Section II, to be equivalent. This only requires checking that the map $\Gamma_{\mathcal{D}} \circ \Gamma_{\mathcal{E}}$ (see Example 5) is an inductive relator for the monad $TX = \mathcal{D}(X + E)$ (which trivially carries a continuous Σ-algebra structure) respecting operations in Σ. This turned out to be an easy exercise. Details can be found in [9].

IX. Related Work

As mentioned in the Introduction, this is certainly not the first paper about program equivalence for higher-order effectful calculi. Denotational semantics of calculi having this nature, has been studied since Moggi's seminal work [30], thus implicitly providing a notion of equivalence. All this has been given a more operational flavour starting with Plotkin and Power account on adequacy for algebraic effects [34], from which the operational semantics presented in this paper is greatly inspired. The literature also offers abstract accounts on logical relations for effectful calculi. The first of them is due to Goubault-Larrecq, Lasota and Nowak [18], which is noticeably able to deal with nondeterministic and probabilistic effects, but also with dynamic name creation, for which applicative bisimilarity is known to be unsound. Another piece of work which is related to ours is due to Johann, Simpson, and Voigtländer [20], who focused on algebraic effects and observational equivalence, and their characterisation via CIU theorems and a form of logical relation based $\top\top$-lifting. In both cases, the target language is typed. Similar in spirit to our approach (which is based on the notion of relator), the work of Katsumata and Sato [22] analyses monadic lifting of relations in the context of $\top\top$-lifting.

Although no abstract account exists on applicative coinductive techniques for calculi with algebraic effects, some work definitely exists in some specific cases. As a noticeable example, the works by Ong [32] and Lassen [25] deal with nondeterminism, and establish soundness in all relevant cases, although full abstraction fails. The first author, together with Alberti, Crubillé and Sangiorgi [10], [8] have studied the probabilistic case, where full abstraction can indeed be obtained if call-by-value evaluation is employed.

X. Conclusion

This is the first abstract account on applicative bisimilarity for calculi with effects. The main result is an abstract soundness theorem for a notion of applicative similarity which can be naturally defined as soon as a monad and an associated relator are given which on the one hand serve to give an operational semantics to the algebraic operations, and on the other need to satisfy some mild conditions in order for similarity to be a precongruence. Soundness of bisimilarity is then obtained as a corollary. Many concrete examples are shown to fit into the introduced axiomatics. A notable example is the output monad, for which a definition of applicative similarity based on labeled transition systems as in e.g. [7] is unsound, a fact that the authors discovered after noticing the anomaly, and not vice versa. Nevertheless, it is possible to define a different notion of applicative similarity that fits into our framework and whose associated notion of bisimilarity (Definition 22) coincide with the usual notion of bisimilarity. The reader can consult [9] for details.

A question that we have not addressed in this work, but which is quite natural, is whether an abstract full-abstraction result could exist, analogously to what, e.g., Johann, Simpson, and Voigtländer obtained for their notion of logical relation.

This is a very interesting topic for future work. It is however impossible to get such a theorem without imposing some further, severe, constraints on the class of effects (i.e. monads and relators) of interest, e.g., applicative bisimilarity is well-known not to be fully-abstract in calculi with nondeterministic effects, which perfectly fit in the picture we have drawn in this paper. A promising route towards this challenge would be to understand which class of *tests* (if any) characterise applicative bisimilarity, depending on the underlying monad and relator, this way generalising results by van Breugel, Mislove, Ouaknine and Worrell [40] or Ong [8].

Finally, environmental bisimilarity is known [23] to overcome the limits of applicative bisimilarity in presence of information hiding. Studying the applicability of the methodology developed in this work to environmental bisimilarity is yet another interesting topic for future researches, to which recent work by Sangiorgi and Vignudelli [37] seems to point naturally.

Acknowledgment

The authors would like to thank Raphaëlle Crubillé and the anonymous reviewers for the many useful comments, some of which led to a substantial improvement of our work. Special thanks go to Davide Sangiorgi, Ryo Tanaka, and Valeria Vignudelli for many insightful discussions about the topics of this work. The first and second authors are partially supported by the ANR projects 12IS02001 PACE, 14CE250005 ELICA, and 16CE250011 REPAS, and by the INRIA-JSPS joint project CRECOGI.

References

[1] Samson Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–117. Addison Wesley, 1990.

[2] Andrew W. Appel and David A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, 2001.

[3] Hendrik P. Barendregt. *The lambda calculus: its syntax and semantics*. Studies in logic and the foundations of mathematics. North-Holland, 1984.

[4] M. Barr. Relational algebras. *Lect. Notes Math.*, 137:39–55, 1970.

[5] Nick Benton, Andrew Kennedy, Lennart Beringer, and Martin Hofmann. Relational semantics for effect-based program transformations: higher-order store. In *Proc. of PPDP 2009*, pages 301–312, 2009.

[6] Ales Bizjak and Lars Birkedal. Step-indexed logical relations for probability. In *Proc. of FOSSACS 2015*, pages 279–294, 2015.

[7] Roy L. Crole and Andrew D. Gordon. Relating operational and denotational semantics for input/output effects. *Mathematical Structures in Computer Science*, 9(2):125–158, 1999.

[8] Raphaëlle Crubillé and Ugo Dal Lago. On probabilistic applicative bisimulation and call-by-value λ-calculi. In *Proc. of ESOP 2014*, volume 8410 of *LNCS*, pages 209–228. Springer, 2014.

[9] Ugo Dal Lago, Francesco Gavazzo, and Paul Levy. Effectful applicative bisimilarity: Monads, relators, and Howe's method (long version). Available at http://arxiv.org/abs/1704.04647, 2017.

[10] Ugo Dal Lago, Davide Sangiorgi, and Michele Alberti. On coinductive equivalences for higher-order probabilistic functional programs. In *Proc. of POPL 2014*, pages 297–308, 2014.

[11] Ugo Dal Lago and Margherita Zorzi. Probabilistic operational semantics for the lambda calculus. *RAIRO - Theor. Inf. and Applic.*, 46(3):413–450, 2012.

[12] Vincent Danos and Russell Harmer. Probabilistic game semantics. *ACM Transactions on Computational Logic*, 3(3):359–382, 2002.

[13] Brian A. Davey and Hilary A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 1990.

[14] Ugo de'Liguoro and Adolfo Piperno. Non deterministic extensions of untyped lambda-calculus. *Inf. Comput.*, 122(2):149–177, 1995.

[15] Joseph A. Goguen, James W. Thatcher, Eric G. Wagner, and Jesse B. Wright. Initial algebra semantics and continuous algebras. *J. ACM*, 24(1):68–95, 1977.

[16] Sergey Goncharov and Lutz Schröder. A relatively complete generic hoare logic for order-enriched effects. In *Proc. of LICS 2013*, pages 273–282. IEEE Computer Society, 2013.

[17] Andrew D. Gordon. A tutorial on co-induction and functional programming. In *Workshops in Computing*, pages 78–95. Springer London, September 1994.

[18] Jean Goubault-Larrecq, Slawomir Lasota, and David Nowak. Logical relations for monadic types. *Mathematical Structures in Computer Science*, 18(6):1169–1217, 2008.

[19] Douglas J. Howe. Proving congruence of bisimulation in functional programming languages. *Inf. Comput.*, 124(2):103–112, 1996.

[20] Patricia Johann, Alex Simpson, and Janis Voigtländer. A generic operational metatheory for algebraic effects. In *Proc. of LICS 2010*, pages 209–218. IEEE Computer Society, 2010.

[21] Claire Jones. *Probabilistic Non-determinism*. PhD thesis, University of Edinburgh, 1990. Available as Technical Report CST-63–90.

[22] Shin-ya Katsumata and Tetsuya Sato. *Preorders on Monads and Coalgebraic Simulations*, pages 145–160. Springer, 2013.

[23] Vasileios Koutavas, Paul Blain Levy, and Eijiro Sumii. From applicative to environmental bisimulation. *Electr. Notes Theor. Comput. Sci.*, 276:215–235, 2011.

[24] Søren B. Lassen. Relational reasoning about contexts. In Andrew D. Gordon and Andrew M. Pitts, editors, *Higher Order Operational Techniques in Semantics*, pages 91–136. 1998.

[25] Søren B. Lassen. *Relational Reasoning about Functions and Nondeterminism*. PhD thesis, Dept. of Computer Science, University of Aarhus, May 1998.

[26] Paul Blain Levy. Infinitary Howe's method. *Electr. Notes Theor. Comput. Sci.*, 164(1):85–104, 2006.

[27] Paul Blain Levy. Similarity quotients as final coalgebras. In *Proc. of FOSSACS 2011*, volume 6604 of *LNCS*, pages 27–41, 2011.

[28] Paul Blain Levy, John Power, and Hayo Thielecke. Modelling environments in call-by-value programming languages. *Inf. Comput.*, 185(2):182–210, 2003.

[29] Saunders MacLane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

[30] Eugenio Moggi. Computational lambda-calculus and monads. In *Proc. of (LICS 1989*, pages 14–23. IEEE Computer Society, 1989.

[31] J. Morris. *Lambda Calculus Models of Programming Languages*. PhD thesis, MIT, 1969.

[32] C.-H. Luke Ong. Non-determinism in a functional setting. In *Proc. of LICS 1993*, pages 275–286. IEEE Computer Society, 1993.

[33] Andrew M. Pitts. Howe's method for higher-order languages. In D. Sangiorgi and J. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*, volume 52 of *Cambridge Tracts in Theoretical Computer Science*, chapter 5, pages 197–232. Cambridge University Press, November 2011.

[34] Gordon D. Plotkin and John Power. Adequacy for algebraic effects. In *Proc. of FOSSACS 2001*, pages 1–24, 2001.

[35] Gordon D. Plotkin and John Power. Notions of computation determine monads. In *Proc. of FOSSACS 2002*, pages 342–356, 2002.

[36] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.

[37] Davide Sangiorgi and Valeria Vignudelli. Environmental bisimulations for probabilistic higher-order languages. In *Proc. of POPL 2016*, pages 595–607, 2016.

[38] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[39] Albert Marchienus Thijs et al. *Simulation and fixpoint semantics*. Rijksuniversiteit Groningen, 1996.

[40] Franck van Breugel, Michael W. Mislove, Joël Ouaknine, and James Worrell. Domain theory, testing and simulation for labelled markov processes. *Theor. Comput. Sci.*, 333(1-2):171–197, 2005.